HeatShield: a low-cost didactic device for control education simulating 3D printer heater blocks

Gergely Takács*, Martin Gulan, Juraj Bavlna, Richard Köplinger, Michal Kováč,
Erik Mikuláš, Sohaibullah Zarghoon and Richard Salíni
Institute of Automation, Measurement and Applied Informatics
Faculty of Mechanical Engineering
Slovak University of Technology in Bratislava
Bratislava, Slovakia
* Corresponding author: gergely.takacs@stuba.sk

***Abstract*—This paper proposes a novel open-source didactic device for control systems engineering education that implements the thermal control of a 3D printer heating block as the underlying dynamic process. The teaching aid is built on a printed circuit board copying the physical outline and the standardized electrical connections of the Arduino Uno microcontroller prototyping board. The educational tool described here can be manufactured for a diminutive cost, thus is accessible to students as a take-home experiment. Moreover, it conforms to the design philosophy of the so-called Arduino Shields, thus, it extends the inherent capabilities of the baseboard and may be combined with other available shields. The proposed hardware is augmented by an application programming interface available in C/C++, MATLAB and Simulink. The paper describes the use of this interface and suggests a sample exercise for the grey-box identification of the differential equation representing the system dynamics which was derived by a first-principles approach. The article also acquaints the reader with sample exercises implementing PID control using the various available programming interfaces.**

***Index Terms*—control engineering education, microcontrollers, Arduino, three-dimensional printing, student experiments, low cost, take-home experiments**

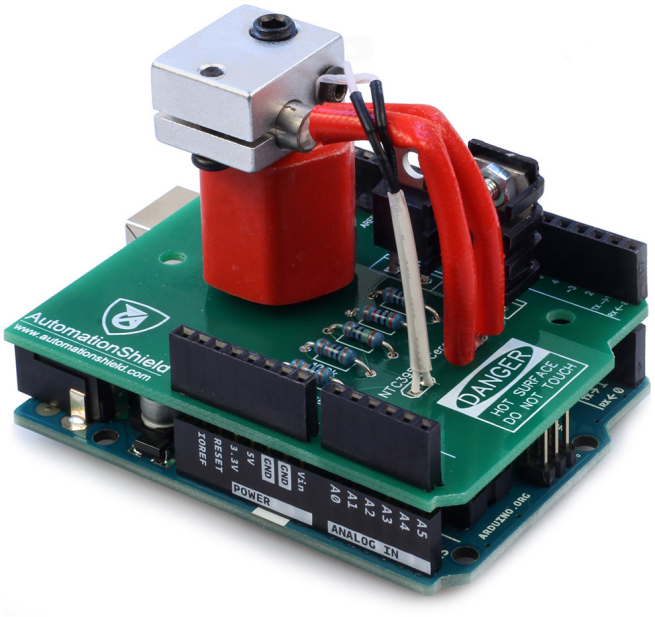


Fig. 1. The open-source HeatShield control systems engineering and mechatronics teaching aid mounted to an Arduino Uno.

I. Introduction

Control systems engineering education has changed immensely since the introduction of comprehensive software tools such as MATLAB or LabView. Problem solving by pen and paper has been reinforced and extended by the computer-aided design and simulation of feedback systems. However, the highly competitive job market places additional expectations on students of control and mechatronics engineering, their instructors and education institutions. Future employees are often expected to be readily capable of implementing real-time feedback control algorithms on hardware. One common target for control algorithms are embedded devices employing microcontroller units (MCU) to compute the control action.

The arrival of the Arduino MCU prototyping platform sparked a new renaissance of hobbyist electronics. It provides an easy but powerful introduction to the world of programmable electronics for students and professionals alike. The considerable success of Arduino rests on the open-source nature of its hardware and the accompanying integrated development environment (IDE), low price and availability, a supportive community of enthusiasts and an immense hardware and software ecosystem.

One aspect of the success of Arduinos is mentioned less frequently: standardization. Hardware-wise, the physical layout and electrical connection of the pins derived from the Arduino Uno, currently in the third revision (R3), is kept constant even across products with various capabilities. This makes the design and production of extension boards known in the Arduino taxonomy as 'Shields' virtually effortless. Along with the pin layout, the Arduino IDE provides a straightforward way to create embedded program applications. The default software libraries and slight modifications to the standard

The authors gratefully acknowledge the contribution of the Slovak Research and Development Agency (APVV) under the contract APVV-14-0399. The authors appreciate the financial support provided by the Cultural and Educational Grant Agency (KEGA) under the contract 005STU-4/2018.

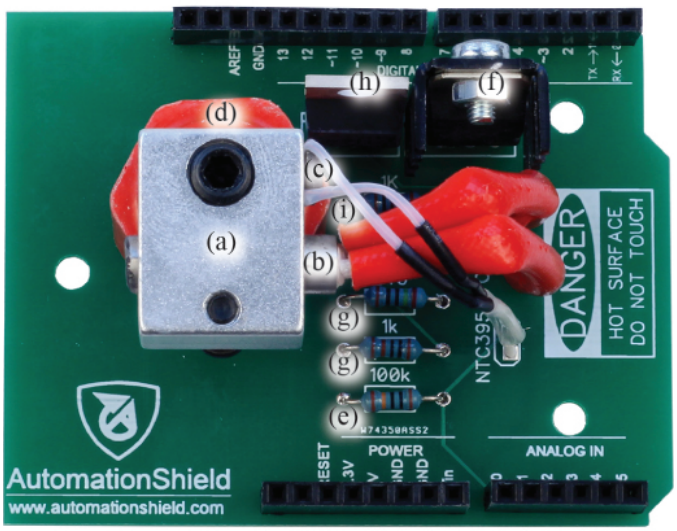


Fig. 2. Top view of the device showing the heating block and the power electronics. See Sect. II-A for description.

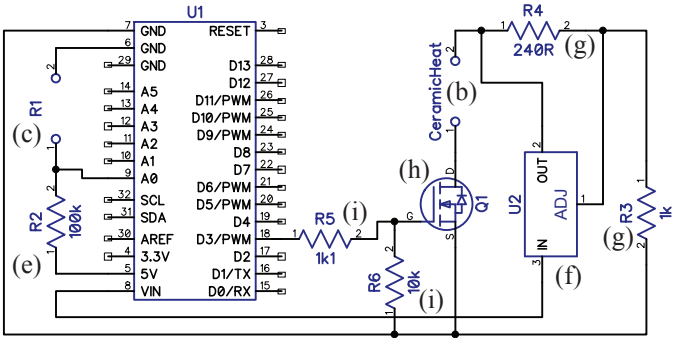


Fig. 3. Electrical schematics of the proposed device.

compilation toolchain specific to the Arduino IDE form a certain type of dialect for the C/C++ language. Even some professional program development environments now offer compilation of Arduino dialect C/C++ directly.

Naturally, Arduinos have penetrated not only the world of makers and hobbyists but academic institutions of higher education as well [14], [16]. The prototyping board in itself may augment standard computer science and programming classes. Yet, its true value in engineering education shows when it is used in conjunction with external hardware. Arduinos augmented by purpose-built educational aids are used in the teaching of electronic engineering [23], signal processing [10], microcontroller technology [18] or, e.g. biomedical engineering concepts [21].

The field of control systems engineering and mechatronics is also a prime target for implementing Arduino-based hardware into the curriculum. The predominant majority of articles discussing the use of Arduinos do not propose educational aids or didactic instruments, instead describe makeshift experiments and sample exercises, e.g. [2], [13], [25], [29]. In other words, Arduinos only act as a low-cost alternative to more expensive laboratory data acquisition (DAQ) cards. Current literature is also abundant with authors proposing various robotic devices built around these prototyping boards, c.f. [6], [11], [15].

Educational aids for teaching feedback control concepts in automation are somewhat less represented. The most common experiment used in conjunction with Arduinos are the ubiquitous motors [1], [4], [7], [9], [24]. There are examples of feedback control experiments with thermal plants [6], RC-circuits [9], optical systems [27], the well-known inverted pendulum [3] and ball-on-plate experiments [5], CNC plotters [20] or, for example, airflow-controlled flaps [17], [19], [25].

Unfortunately, most of these devices contain improvised mechanical components, custom enclosures or base structures. Their construction details most often remain hidden or are insufficiently documented in the original papers. The underlying dynamic process and idea may be universal, but the presented hardware is a one-off prototype. Even if the hardware itself is open-source, the production price is still usually in the range of $100–$150 and may require specialized equipment. These didactic tools for feedback control are therefore unlikely to permeate into curriculums outside of their creators' classes. Sadly, this also means that a crowd-sourced improvement of the hardware, software libraries or even exercises—like in the case of the Arduino itself—is not possible.

The alternative to this approach in teaching feedback control concepts is to use commercial didactic tools. Regrettably these instruments are large, delicate and most importantly prohibitively expensive for some universities and most students. The changes in education practices are obvious, the so-called take-home laboratories would allow educators to assign experiment-oriented homework and individual projects [22].

Certain authors have embraced the idea of building didactic devices for Arduinos configured in the framework of the aforementioned 'Shields'. For example, Sarik and Kymissis have presented a laboratory kit for electronic engineering [23]. Recently Dočekal and Golembiovský have integrated two serially connected RC circuits into a low-cost Arduino Shield to teach control engineering principles [9]. The authors of this article have introduced a simple optical feedback experiment for control education that is essentially an Arduino shield and may be manufactured under ∼$3 [27].

Candelas et al. recommended a student experiment to control the temperature in the extrusion head of 3D printers commonly known as the 'hot-end' [6]. The authors created an ad-hoc demonstration device with improvised parts that attached to the Arduino externally. The experiment was configured so that the Arduino switched a relay, effectively constituting a two-state control feedback loop. This article introduces a didactic device for teaching feedback control concepts based on this underlying process, however, with an open-source hardware and software design that is compliant with the Arduino Shield principle.

TABLE I
COMPONENT LIST OF THE PROPOSED DEVICE WITH PRICES IN USD.

Description	Designator	Value	Quantity	Unit Price[a]	Price
Transistor	Q1	BUZ11, N-channel power MOSFET, 50V, TO-220 THT	1	$0.262	$0.262
Voltage regulator	U2	LM317T, Adjustable positive linear voltage regulator, 1.2–37 V, TO-220, THT	1	$0.116	$0.116
Heat sink	—	Heatsink for the voltage regulator, TO-220 package	1	$0.053	$0.053
Thermistor	R1	NTC 3950, 100 kΩ ± 1%, rated 300 °C with wiring	1	$0.445	$0.445
Resistor	R2	100 kΩ, 1/4 W, THT	1	$0.016	$0.016
Resistor	R3, R5	1 kΩ, 1/4 W, THT	2	$0.016	$0.032
Resistor	R4	240 kΩ, 1/4 W, THT	1	$0.016	$0.016
Heating cartridge	—	24V, 30W, 20×6 mm	1	$0.98	$0.98
Heater block	—	Aluminum, 20×16×12 mm	1	$0.99	$0.99
Bolt		M6×20 mm, headless, block to insulator	1	0.0526	0.0526
Thermal insulator	—	Hexagonal glass-filled polyester insulator M6 IS20HH625, 25×20 mm	1	$0.89	$0.89
Bolt	—	M6×10 mm, rounded 3.3 mm flat head, insulator to PCB	1	$0.10	$0.10
Header	—	6x1, female, 2.54 mm pitch	1	$0.070	$0.070
Header	—	8x1, female, 2.54 mm pitch	2	$0.099	$0.198
Header	—	8x1, female, 2.54 mm pitch	1	$0.099	$0.099
PCB	—	FR4, 2 layer, 1.6 mm thick	1	$0.50	$0.50
Total:					**$4.83**[b]

[a] For low quantity orders.
[b] Excluding labor and postage.

Like in the paper of Candelas et al. [6], our device implements a heating block from 3D printers encompassing a resistive heating cartridge and a thermistor; thereby creating a single-input, single-output (SISO) process. However, here the entire experimental apparatus is mounted onto a single circuit board that follows the pin layout and outline of the Arduino Uno, thereby conforming to the concept of Arduino Shields (see Fig. 1). We show that the underlying hardware can be produced from commercially available components within the framework of open-source hardware. As we will demonstrate later, this feedback experiment-on-a-board may be manufactured under mere $5, thereby rendering it extremely affordable to students and by that attribute even suitable for homework assignments or long-term projects.

In addition to the thermal experiment-on-a-shield that we chose to call 'HeatShield' in the upcoming discussion, we will also introduce a software interface to simplify and support the learning process along with the worked example code. The paper thus details the application programming interface (API) for various usage scenarios and recommends sample experiments using the proposed hardware.

II. HARDWARE

A. Electrical design

The device consists of a heating element as an actuator and a temperature sensor is used to generate feedback signal. The hardware description of the proposed device is illustrated by the photograph showing the physical arrangement of the components on the board in Fig. 2 and an electrical schematic in Fig. 3. Both figures highlight corresponding components by letters in the upcoming text.

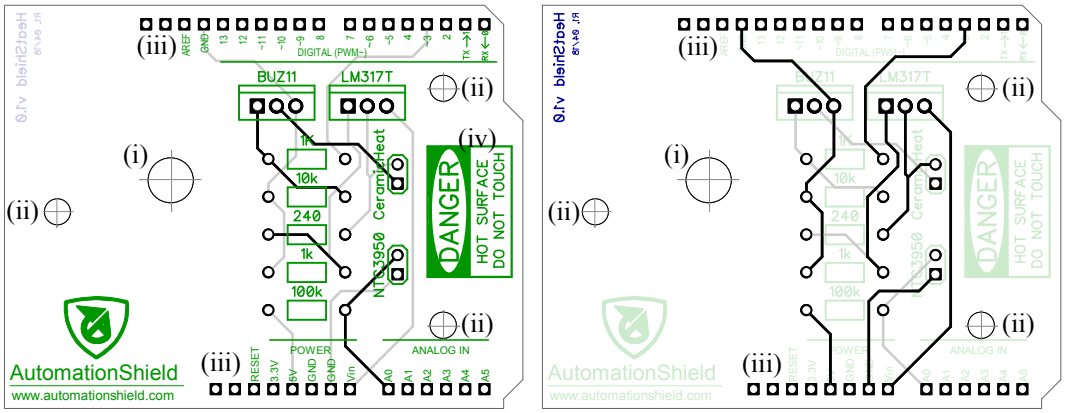
The heating block of the 3D printer that encompasses the plastic extruding module is a 20×16×12 mm aluminum block (a). This block contains the ceramic heating cartridge (b) rated for a maximum of 24 V and 30 W measuring 20×12 mm. Temperature feedback is based on a negative temperature coefficient (NTC) resistor (c). The assembled heating block is mounted on a thermal insulator (d), in order to prevent heat damage to the PCB. In place of the extruding head supplying the melted plastic filament is a steel screw connecting the block to the insulator.

The thermistor with a nominal 100 kΩ resistance is connected to the A0 analog input of the Arduino in a voltage divider circuit paired with another 100 kΩ resistor (e). Normal temperatures used to melt plastics in 3D printing are over 200 °C and may reach as high as 320 °C for polycarbonate (PC) filaments. This, unfortunately, would not be safe for general classroom use. The maximal temperature of the heating block was therefore maximized at ∼80 °C . Because thermal exposure of skin is a combination of temperature and time [8], at this heating level the critical exposure is less than a second. While this can still pose a certain risk of injury if accidently touched, still enough time is left to react and withdraw a finger.

Although it would be possible to limit the temperature of the printing head by software in the API only, we chose to implement a hardware solution for safety reasons. The maximal possible voltage and therefore current and heating power supplied to the heating cartridge is limited by an adjustable linear voltage regulator (f). This voltage regulator takes the external 12 V input from the Vin pin and reduces it to a voltage configured by resistors (g) supplying its adjustable input.

The power circuitry is driven by an N-channel metal-oxide semiconductor field-effect transistor (MOSFET) (h). This transistor is connected to the pulse-width modulation (PWM) capable D3 pin of the Arduino. A series resistor protects the MCU in transients, while a parallel to ground ensures that floating electrical states do not cause the heater to turn off accidentally (i).

The exact component specification and required quantities are summarized in Table I. Only components with through-hole technology mounting (THT) have been chosen in or-



(a) Top layer. (b) Bottom layer.

Fig. 4. Printed circuit board with a 1:1 scale of the proposed device showing traces and pads (black), drilled holes and the silkscreen layers (green, blue).

der to make assembly and servicing easy even for students inexperienced with electronics. The table shows component pricing for low quantity orders that excludes labor price and postage. According to this summary, the proposed device should cost no more than $5, whilst the manufacturing price can be significantly reduced for high quantity production.

B. Circuit board

The printed circuit board (PCB) of the proposed device is illustrated in Fig. 4. First and foremost, the PCB accommodates a hole (i) to mount the insulator with the heating block on top. The board also incorporates three drilled holes (ii) allowing one to mount the 3D-printed safety enclosure. The placement of the stackable female headers (iii) copies the physical and electrical pin layout of the Arduino Uno. Pins and components are marked by the silkscreen layer, including a warning sign about the possible risk of injury by the hot surface.

The physical outline of the printed circuit board copies that of the Arduino Uno, which, in conjunction with the headers, creates a standard Arduino shield. Traces have been distributed between two layers and our prototypes have been manufactured as 1.6 mm thick FR4 glass composite panels. The size of the board fits into the usual 100×100 mm limit of most economy PCB manufacturers, reducing the unit price for $0.50 even for small quantity manufacturing.

The circuit schematics were designed in the free version of the DipTrace CAD software. The schematic was then converted into the PCB design. Editable schematic and board design files are freely accessible [26]. In addition to these, the exported manufacturing layers and drill files ready for production are downloadable as well [26].

C. Safety enclosure

Although the probability of a thermal injury from touching the heating block is minimal, an optional 3D printed safety enclosure has been also designed (see Fig. 5). The enclosure has two components; a box-like superstructure and a lid on the top that can be opened to access the internals, if needed. The superstructure is fastened onto the shield by three M3 bolts and corresponding nuts. The open parts of the box and lid are fitted with 2 mm thin Plexiglas.

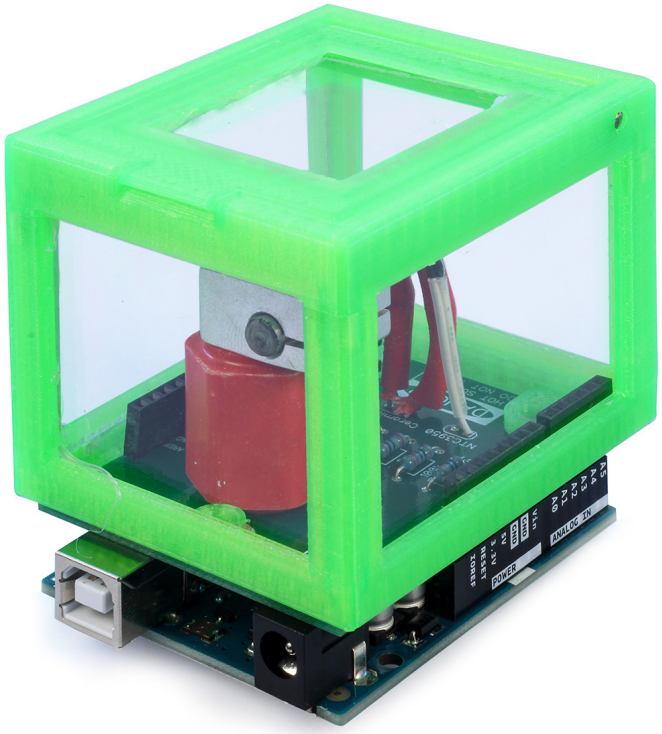


Fig. 5. The 3D printed safety enclosure with Plexiglass sides mounted onto the proposed device.

The 3D CAD files for the enclosure are also available for download [26]. It is recommended for the box be separated into two different parts when printing to prevent deformations. The 3D printing of an enclosure can be realized in a single print cycle in about 2 h 30 min. The enclosure requires 23 g (8.12 m) filament, raising the unit price approximately by $0.55. This, and the price of the required bolts and nuts is still well under $1.0. We prepared the G-code for the enclosure

shown in Fig. 5 in Slic3er, then printed the parts in on an Anet A8 3D printer.

III. Application Programming Interface

Creating the software interface for the thermistor and the heating cartridge may be of didactic value in itself. Such exercises are welcome mainly in courses teaching the principles of embedded system design and electronics.

There is unfortunately no time to work on the interface or other low-level hardware issues in courses focused primarily on control systems engineering or system identification principles. This is especially true with control theory courses. For this reason, we have created an application programming interface to render the tool instantly usable in class.

A. C/C++ library for Arduino IDE

The API serving the device proposed in this particle was written in C/C++ and is integrated into the AutomationShield open-source library for the Arduino IDE [26]. This library contains hardware drivers and sample exercises for control systems engineering education. All functionality associated with the HeatShield is included in the `HeatShield.h` header, this contains the `HeatShieldClass` class that is constructed by default as the `HeatShield` object.

To begin work the students must initialize the hardware by calling

```
HeatShield.begin();
```

this determines the mode of the input and output pins.

The thermistor in the heating block is accessed by calling the

```
y = HeatShield.sensorRead();
```

method, which returns the block temperature in degrees Celsius to the variable `y` as a floating point number. This function first calls the `getThermistorVoltage()` method, which returns the output potential at the voltage divider. Based on the known input reference voltage V_r, the known reference resistance R_r and the the output voltage V_o one may use Kirchhoff's current law to compute the unknown resistance R according to

$$R = \frac{V_\mathrm{o} R_\mathrm{r}}{V_\mathrm{r} - V_\mathrm{o}}; \tag{1}$$

which is implemented in the device API as the `getThermistorResistance()` method. Though searching for the unknown temperature is more precise according to tables associating the nonlinear relationship of resistance and temperature, we decided to rely on the simplified form of the Steinhart-Hart equation. Given a known reference temperature T_0 (K) and corresponding nominal resistance R_0 (Ω); and the properties of the thermocouple given by the parameter β one may compute the unknown temperature T (K) by

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{\beta} \ln\left(\frac{R}{R_0}\right), \tag{2}$$

which is what `sensorRead()` essentially does.

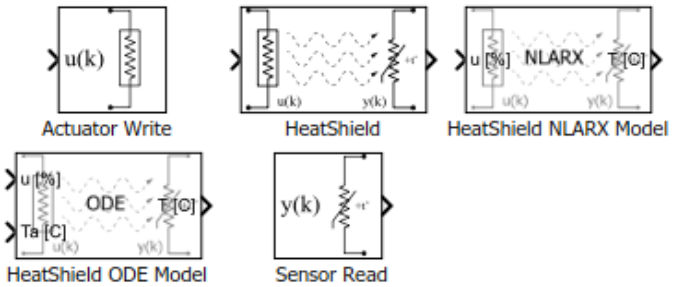


Fig. 6. Algorithmic blocks for the Simulink API of the proposed device.

Assuming the power supplied to the heating cartridge is stored in the variable `u` as a floating point number in the range of 0–100 (%), the actuator is supplied by the input signal after calling the method

```
HeatShield.actuatorWrite(u);
```

which will convert the percentage value to an 8-bit number driving the PWM output of the microcontroller.

B. MATLAB API

The MATLAB Support Package for Arduino Hardware enables communication between the Arduino prototyping platform and the development computer. Various commands accessing the hardware are executed directly in quasi real time without the need to compile code. This is achieved by uploading a server code to the MCU, which is then accessed via the serial communication protocol. It is important to realize that code is not deployed to the processor; the Arduino merely acts as an external laboratory measurement card.

Due to the quasi real-time nature of code execution, fast systems with sample times below $T_\mathrm{s} = 0.1$–0.2 s cannot be reliably run to this interface. However, plants with slow dynamics, such as the proposed device, can be used in this fashion. The ability to access the hardware through the MATLAB command line and scripts is essential for those beginning students who cannot program in C/C++ just yet. The capability to run feedback experiments directly from scripts is also essential in teaching more advanced control engineering courses, where the underlying principle makes use of the high-level commands of MATLAB and would be otherwise complex and time consuming to implement. Such are, for example, courses including model predictive control (MPC) or adaptive control with online system identification features.

In order to prevent confusion between the C/C++ and the MATLAB API, we have attempted to make the two interfaces as similar as possible. The MATLAB API is written in object-oriented script and the user must first create an instance from the class:

```
HeatShield=HeatShield;
```

The shield is then initialized by calling

```
HeatShield.begin();
```

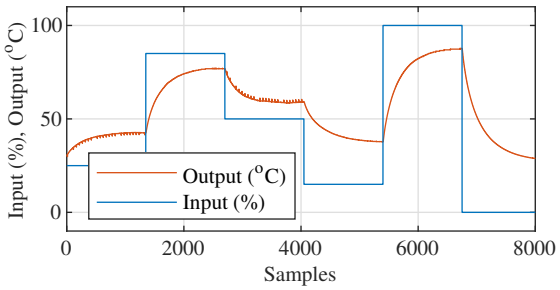


Fig. 7. System identification experiment by a sequence of open-loop step responses implemented through the C/C++ API, logged via the MATLAB API.

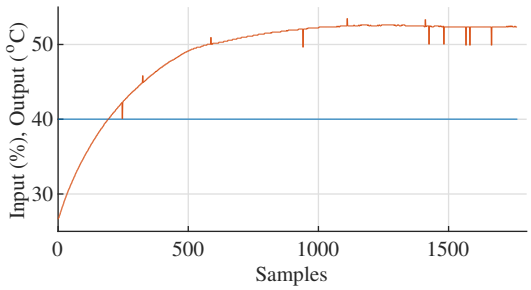


Fig. 8. Live system identification experiment by an open-loop step response implemented through the MATLAB API (blue - input, orange - output).

which, unless already done, uploads the server code to the prototyping board.

The temperature reading procedure is identical to that introduced previously for the C/C++ API. The voltage from the `getThermistorVoltage()` method is passed to the `getThermistorResistance()` and this will calculate the resistance of the thermistor based on the voltage divider. The sensor is then directly accessed by calling the

```
y = HeatShield.sensorRead();
```

method, where the variable `y` will hold the temperature in degrees Celsius. Finally, the heating cartridge power is set through

```
HeatShield.actuatorWrite(u);
```

which accepts the input power `u` in percents. The MATLAB API is also augmented by scripts implementing several examples, see Sect. IV for further discussion.

C. Simulink API

The Simulink API for the proposed hardware utilizes the Simulink Support Package for Arduino Hardware. This, in essence, supplies algorithmic units in blocks that access the hardware functionality. In direct contrast with the way MATLAB handles Arduinos, the block scheme in Simulink is transcribed into C/C++, then compiled to machine code and uploaded to the prototyping board. In other words, code is run directly on the microcontroller. Simulink not only transcribes the block schemes for hardware, it also maintains the connection between the development computer and MCU. This way controllers can be fine-tuned in a live session; or data may be displayed and logged conveniently.

Thus, in addition to the C/C++ and MATLAB API, we have also created a Simulink API for the HeatShield. We found that creating control loops in these inherently time-dependent visually engaging block schemes can be an intuitive and valuable educational experience for students. The live communication using switches, buttons and other built-in interface modules may aid the didactic process as well.

The algorithmic blocks specific to the hardware proposed here are part of the AutomationShield library. These blocks are illustrated in Fig. 6. The 'Actuator Write' block accepts real numbers from 0–100% and supplies power to the cartridge. The 'Sensor Read' block reads the input from the thermistor. Users may choose between raw ADC levels, voltage, resistance and temperature. The parameters of the voltage divider and the thermistor can be also fine-tuned.

The 'HeatShield' block unites the input and output functionality into a single entity that can be conveniently used for identification and control experiments. The dynamic process of heating the printer head is mathematically represented by a black-box nonlinear ARX model and a first-order ordinary differential equation. The blocks can be used to simulate the response of HeatShield and thereby tuning controllers without the need to run the experiments on the live system.

IV. Demonstration Experiments

A. System Identification

Input-output experiments for data gathering can be launched, displayed and logged in C/C++ (Arduino IDE), MATLAB and Simulink as well. For example, one worked example in the C/C++ sends a series of step changes that can be logged in various ways through the serial line, while another also adds a random component to the inputs to collect more information-rich signals (Fig. 7). Similarly, a worked MATLAB example performs a simple step change of input (Fig. 8), displays the progress of the response live on screen and saves a data file.

After the students gather enough data sufficient for system identification, they may be encouraged to discuss the nature of the process, then try to fit a model to the experimental response. By inspecting the response it is clear that the dynamics is of the first order, and one may assume that the identification procedure is straightforward. Nevertheless, the simple dynamics of the HeatShield holds surprises and may be used to illustrate several abstract principles by a real-life example. The MATLAB API for the the proposed device contains a worked example for black-box system identification. This creates a simple but inadequate process model and a more complex and valid nonlinear ARX model. If one assumes a first-order dynamics, where τ is the time constant and K is the DC gain, then the ODE

$$\tau \dot{T}(t) + T(t) = Ku(t) \tag{3}$$

TABLE II
PARAMETER TABLE AND INITIAL GUESS FOR THE HEATSHIELD FIRST-PRINCIPLE MODEL

Parameter	Symbol	Guess	Unit
Weight of the heating block	m	0.01	kg
Specific heat of aluminum	c	897	J· K/kg
Cartridge resistance	R	18	Ω
Printer head surface area	A	1504E-6	m^2
Convective coefficient (air)	h	20	$W{\cdot}K/m^2$
Nominal voltage of PWM signal	V	6.45	V
Ambient temperature	T_a	28	°C

can be easily fit to a generic first-order process model. This hypothesis can be tested by the aforementioned worked example using the System Identification Toolbox, which creates process models from supplied data. Students can quickly find out that this assumption provides a poor fit (∼20%), and the initial idea is wrong.

Some students may continue the identification procedure by detrending the supplied data, thereby assuming the presence of a constant effect—the ambient temperature and convection. Testing this using a process model will yield a much better match. This assumption would suggest the ODE is in the generic form of

$$\tau\dot{T}(t) + T(t) - \alpha = Ku(t), \tag{4}$$

where the constant α can express a dynamic effect due to the ambient temperature and convection. At this point, the dynamics of the proposed device can be used to demonstrate the principle of superposition, since the Eq. (4) is although linear in nature, it is not a linear map of the input to the output (See Appendix A.). This may motivate students to create a nonlinear black-box model, which achieves a much better match to the data (∼90%).

Developing a first-principle model, then fitting the data to this using grey-box methods is an even more sophisticated assignment. This procedure is also implemented as a worked example for the proposed device.

First, let us create an energy balance equation. Any power not lost to the Joule heating process to convection must remain as energy in the heating block itself. Mathematically this is

$$\dot{Q}_C(t) = \dot{Q}_J(t) - \dot{Q}_R(t), \tag{5}$$

where $\dot{Q}_C(t)$ denotes energy change from heat capacity (thermal capacitance), $\dot{Q}_J(t)$ is the Joule heating process through the cartridge while $\dot{Q}_R(t)$ represents energy transfer by convection to the surroundings, that is often referred to by the misnomer thermal resistance in control engineering circles.

Assuming no mass transfer occurs, the change in the heat is given by

$$\dot{Q}_C(t) = mc\dot{T}(t), \tag{6}$$

where m is the mass of the block and c is the mass-specific heat capacity, while $T(t)$ is the temperature of the block. Heat transfer through convection is given as

$$\dot{Q}_R(t) = hA(T(t) - T_a(t)), \tag{7}$$

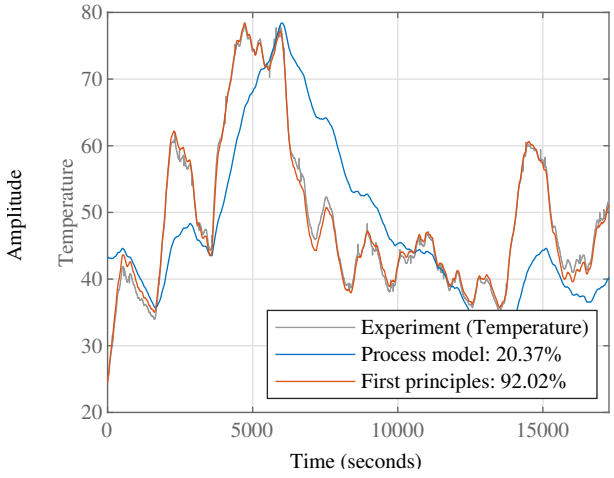


Fig. 9. Simulated response comparison of the first order process model and the first-principle model with measurement data.

where T_a is the ambient temperature, A is the area of the heating block and h is the convective heat transfer coefficient. If we assume a purely resistive heating cartridge with constant resistance, we can compute the power supplied to the block according to Ohm's law as the root mean square of the voltage divided by the resistance. We are also aware of the nature of the incoming signal, that is pulse-width modulated. Based on this, the power supplied to the block is

$$\dot{Q}_J(t) = U(t)I(t) = \frac{U^2_{\mathrm{rms}(t)}}{R} = \left(V\sqrt{\frac{t_o(t)}{t_p}}\right)^2 \frac{1}{R} = \tag{8}$$
$$= \frac{V^2}{R}\frac{t_o(t)}{t_p} = \frac{V^2}{R}\left(\frac{1}{100}u(t)\right),$$

where $U(t)$ is the voltage potential, $I(t)$ is the current and R is the resistance of the coil. Furthermore, t_p is the period, t_o is the active duty time and finally V is the constant voltage amplitude of the supplying PWM signal. The parameters used in these equations, their units and initial guesses are listed in Table II.

Substituting Eq. (6)–(8) to the energy balance equation in (5) yields

$$mc\dot{T}(t) = \frac{V^2}{R}\left(\frac{1}{100}u(t)\right) - hA(T(t) - T_a(t)) \tag{9}$$

from which is clear that our original intuition about the nature of the process expressed by the generic ODE in Eq. (4) was correct.

By isolating the derivative of temperature on the left side of Eq. (10), we can create a first-order state-space representation

$$\dot{T}(t) = \frac{V^2}{mcR}\left(\frac{1}{100}u(t)\right) - \frac{hA}{mc}(T(t) - T_a(t)) =$$
$$= -\frac{hA}{mc}T(t) + \frac{V^2}{mcR}\left(\frac{1}{100}u(t)\right) + \frac{hA}{mc}T_a(t) \tag{10}$$

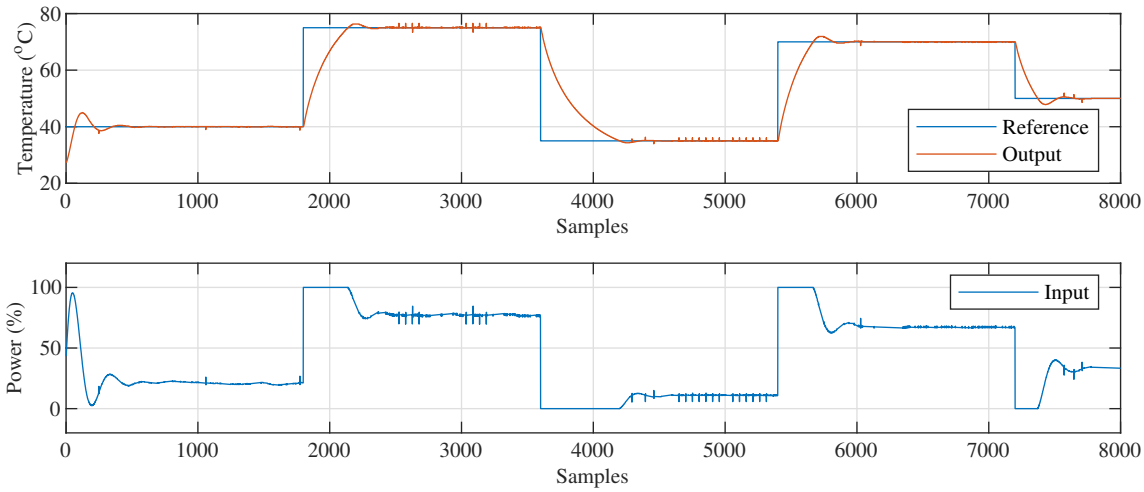


Fig. 10. Closed-loop response of the printer head temperature in the C/C++ API implementation of the PID example, logged in the MATLAB API.

by assuming $x(t) = T(t)$ and $y(t) = x(t)$. This state-space model can be expressed as a MATLAB ODE function, which can be then utilized in a grey-box identification procedure. The ODE model of the process in `HeatShield_ODE.m` is a part of the API.

A worked example takes experimental data and searches for the unknown parameters of the first-principle model. First, an initial guess of the model is created based on the assumed parameter values and model structure. The parameters are then found by a nonlinear grey box estimation procedure using an adaptive Gauss-Newton search method. The resulting model is an excellent match to the measured input-output data (92%). This model is also implemented in the Simulink API.

B. Control

The possible range of student experiments in implementing control algorithms is only limited by imagination and, more pragmatically, the computational load of the chosen method on the AVR processor architecture. From the viewpoint of the sampling period and the processor load this is not an issue, since the slow thermal response of the 3D printer head allows sampling in the range of several seconds. The aspect of limited operational or program memory is not likely to prohibit creativity either. Although the 8-bit microcontroller of the Arduino tends to be underestimated, it has been even used to implement model predictive control in real time [12], [28]. Even if students reach the limits of the MCU, the MATLAB API still provides a way to test complex control and estimation algorithms. Thus, the full high-level mathematic might of MATLAB may be utilized when exploiting the provided API.

Here we have merely chosen to demonstrate closed-loop temperature control by the PID algorithm, as this is likely to be part of any and all undergraduate control curricula. The implementation of PID control in C/C++ is demonstrated by a worked example (`HeatShield_PID.ino`). This example

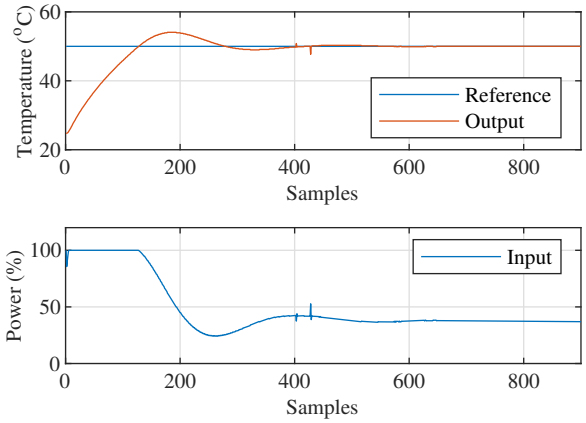


Fig. 11. Live PID control experiment in the MATLAB API.

makes use of the interrupt-driven sampling subsystem of the AutomationShield library, and also its built-in input-saturated absolute-form PID methods with integral windup handling by clamping. The progress of the experiments can be followed in real time through the Serial Plotter of the Arduino IDE, here we have chosen to log the response in MATLAB. The results of the PID controlled temperature response of the printer head are shown in Fig. 10. The same experiment can be conveniently launched from the MATLAB API as well, see Fig. 11. Meaningful experiments such as the one illustrated here can be conducted in less than 30 minutes of time.

Here we shall also illustrate the ease of building control loops for the proposed device in Simulink. Figure 12 shows the full block scheme for discrete saturated PID control of the proposed device. Students need only to select the 'HeatShield' block from the API to implement the input/output of the hardware. The rest of the blocks—such as the 'Discrete PID

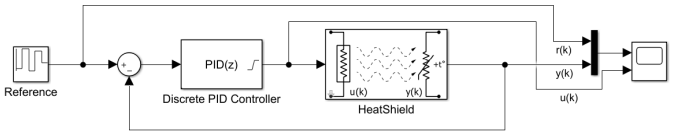


Fig. 12. Block scheme for discrete saturated PID control using the HeatShield block from the Simulink API.

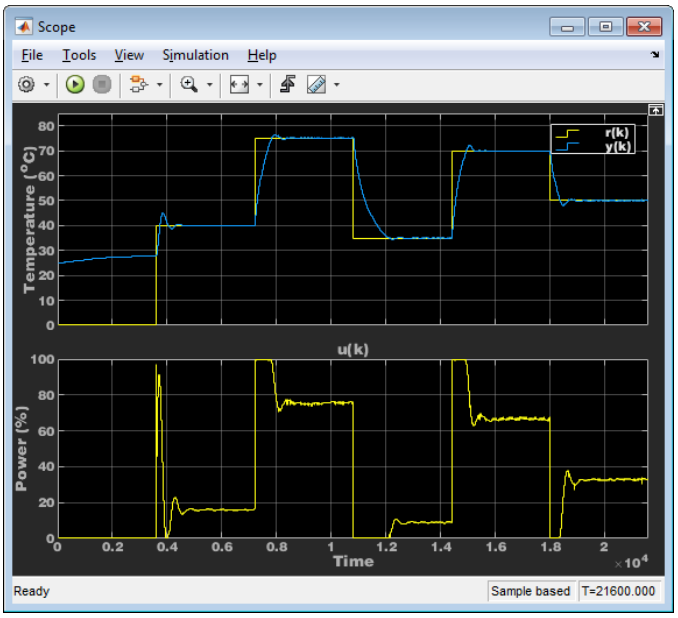


Fig. 13. Example screenshot of the Simulink Scope output, showing live data from PID control.

Controller'—can be readily selected from the default library.

After selecting an External running mode the block scheme is re-interpreted to C/C++ code, which is then compiled to AVR-specific machine code and downloaded to the MCU. Communication is possible between the block scheme and the hardware. Students may use switches, sliders and knobs to select reference levels and inspect the response live using a Scope (see Fig. 13).

V. Conclusion

A novel educational aid for teaching control systems engineering and mechatronics has been presented in this paper. The proposed hardware simulating the thermal processes in a 3D printer heating block can be manufactured for as low as $5, making it ideal for practical take-home student experiments. Since the API is available in C/C++, MATLAB and Simulink as well; instructors are granted a flexible choice when creating assignments. In addition to the standard fare of classical control schemes, the device can also be used to teach e.g. concepts of optimal control and estimation, due to the slow dynamics of the underlying thermal process.

Future revisions of the hardware may include another thermocouple monitoring the ambient temperature, thus making the modeling and identification of the process more precise and engaging. The PCB can also accommodate a potentiometer, the role of which can be determined by the students, for example it may be used to set the reference temperature. Furthermore, adding a fan or mounting a Peltier device for thermoelectric cooling could either add another input to the system or act as a repeatable way to inject disturbance into the closed-loop control process.

References

[1] R. O. Aldeyturriaga, C. A. Junior, A. S. Silveira, and A. A. Coelho. Low cost setup to support pid ideas in control engineering education. *IFAC Proceedings Volumes*, 46(17):19 – 24, 2013. 10th IFAC Symposium Advances in Control Education.

[2] K. Asato, T. Nagado, and S. Tamaki. Development of low cost educational material for learning fundamentals of mechatronics. In *2015 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, pages 454–456, Nov 2015.

[3] P. Bakaráč, M. Kalúz, and Ľ. Čirka. Design and development of a low-cost inverted pendulum for control education. In M. Fikar and M. Kvasnica, editors, *Proceedings of the 21st International Conference on Process Control*, pages 398–403, Štrbské Pleso, Slovakia, June 6-9, 2017 2017. Slovak University of Technology in Bratislava, Slovak Chemical Library.

[4] R. Barber, M. Horra, and J. Crespo. Control practices using Simulink with Arduino as low cost hardware. *IFAC Proceedings Volumes*, 46(17):250 – 255, 2013. 10th IFAC Symposium Advances in Control Education.

[5] C. J. Bay and B. P. Rasmussen. Exploring controls education: A reconfigurable ball and plate platform kit. In *2016 American Control Conference*, pages 6652–6657, July 2016.

[6] F. Candelas, G. García, S. Puente, J. Pomares, C. Jara, J. Pérez, D. Mira, and F. Torres. Experiences on using Arduino for laboratory experiments of automatic control and robotics. *IFAC-PapersOnLine*, 48(29):105 – 110, 2015. IFAC Workshop on Internet Based Control Education IBCE15.

[7] B. Çatalbaş and İsmail Uyanık. A low-cost laboratory experiment setup for frequency domain analysis for a feedback control systems course. *IFAC-PapersOnLine*, 50(1):15704 – 15709, 2017. 20th IFAC World Congress.

[8] M. W. Dewhirst, B. L. Viglianti, M. Lora-Michiels, P. J. Hoopes, and M. Hanson. Thermal dose requirement for tissue effect: Experimental and clinical findings. *Proceedings of SPIE–the International Society for Optical Engineering*, 4954:37, 2003.

[9] T. Dočekal and M. Golembiovsky. Low cost laboratory plant for control system education. *IFAC-PapersOnLine*, 51(6):289 – 294, 2018. 15th IFAC Conference on Programmable Devices and Embedded Systems PDeS 2018.

[10] W. J. Esposito, F. A. Mujica, D. G. Garcia, and G. T. A. Kovacs. The lab-in-a-box project: an Arduino compatible signals and electronics teaching system. In *2015 IEEE Signal Processing and Signal Processing Education Workshop (SP/SPE)*, pages 301–306, Aug 2015.

[11] C. Gonzalez, I. Alvarado, and D. Muñoz La Peña. Low cost two-wheels self-balancing robot for control education. *IFAC-PapersOnLine*, 50(1):9174 – 9179, 2017. 20th IFAC World Congress.

[12] M. Gulan, G. Takács, N. A. Nguyen, S. Olaru, P. Rodriguez-Ayerbe, and B. Rohal'-Ilkiv. Embedded linear model predictive control for 8-bit microcontrollers via convex lifting. *IFAC-PapersOnLine*, 50(1):10697–10704, 2017. 20th IFAC World Congress.

[13] R. C. Hill. Hardware-based activities for flipping the system dynamics and control curriculum. In *2015 American Control Conference*, pages 2777–2782, July 2015.

[14] J. Hurtuk, M. Chovanec, and N. Ádam. The Arduino platform connected to education process. In *2017 IEEE 21st International Conference on Intelligent Engineering Systems*, pages 71–76, Oct 2017.

[15] M. Ishikawa and I. Maruta. Rapid prototyping for control education using Arduino and open-source technologies. *IFAC Proceedings Volumes*, 42(24):317–321, 2010. 8th IFAC Symposium on Advances in Control Education.

[16] P. Jamieson and J. Herdtner. More missing the boat — arduino, raspberry pi, and small prototyping boards and engineering education needs them. In *2015 IEEE Frontiers in Education Conference (FIE)*, volume 00, pages 1–6, Oct. 2015.

[17] M. Kalúz, Ľ. Čirka, and M. Fikar. Flexy: An open-source device for control education. In A. Cardoso, editor, *13th APCA International Conference on Automatic Control and Soft Computing*, pages 37–42, Univesrity of the Azores, Ponta Delgada, Portugal, June 4-6 2018. APCA, Nova Gráfica.

[18] J. C. Martínez-Santos, O. Acevedo-Patino, and S. H. Contreras-Ortiz. Influence of arduino on the development of advanced microcontrollers courses. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, 12(4):208–217, Nov 2017.

[19] S. C. McLoone and J. Maloco. A cost-effective hardware-based laboratory solution for demonstrating PID control. In *2016 UKACC 11th International Conference on Control (CONTROL)*, pages 1–6, Aug 2016.

[20] J. Oravec, M. Kalúz, P. Bakárač, and M. Bakošová. Improvements of educational process of automation and optimization using 2d plotter. *IFAC-PapersOnLine*, 49(6):16 – 21, 2016. 11th IFAC Symposium on Advances in Control Education ACE 2016.

[21] S. Puente, A. Úbeda, and F. Torres. e-health: Biomedical instrumentation with arduino. *IFAC-PapersOnLine*, 50(1):9156 – 9161, 2017. 20th IFAC World Congress.

[22] J. Rossiter, S. Dormido, L. Vlacic, B. L. Jones, and R. Murray. Opportunities and good practice in control education: a survey. *IFAC Proceedings Volumes*, 47(3):10568 – 10573, 2014. 19th IFAC World Congress.

[23] J. Sarik and I. Kymissis. Lab kits using the Arduino prototyping platform. In *2010 IEEE Frontiers in Education Conference*, pages T3C–1–T3C–5, Oct 2010.

[24] J. Sobota, R. Pišl, P. Balda, and M. Schlegel. Raspberry pi and arduino boards in control education. *IFAC Proceedings Volumes*, 46(17):7 – 12, 2013. 10th IFAC Symposium Advances in Control Education.

[25] B. Stark, Z. Li, B. Smith, and Y. Chen. Take-home mechatronics control labs: a low-cost personal solution and educational assessment. *ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference Volume 4: 18th Design for Manufacturing and the Life Cycle Conference; 2013 ASME/IEEE International Conference on Mechatronic and Embedded Systems and Applications*, pages 1–9, Aug 2015.

[26] G. Takács, M. Gulan, J. Bavlna, R. Köplinger, M. Kováč, E. Mikuláš, S. Zarghoon, and R. Salíni. Heatshield. Online, 2018. [cited 09.10.2018]; GitHub Wiki page. Available from https://github.com/gergelytakacs/AutomationShield/wiki/HeatShield.

[27] G. Takács, T. Konkoly, and M. Gulan. Optoshield: A low-cost tool for control and mechatronics education. In *Proceedings of the 12th Asian Control Conference (ASCC2019), June 9-12, 2019, Kitakyusu, Japan.*, pages 1–6, Kitakyushu, Japan, Jun 2019. Preprint submitted for review.

[28] G. Takács, P. Zometa, R. Findeisen, and B. Rohal'-Ilkiv. Embedded model predictive vibration control on low-end 8-bit microcontrollers via automatic code generation. In *ICSV 23: Proceedings of the 23rd International Congress on Sound and Vibrattion. Athens, Greece, 10-14 July, 2016*, pages 266/1–266/8, Athens, Greece, July 2016.

[29] C. Yfoulis, S. Papadopoulou, D. Trigkas, and S. Voutetakis. Switching pi speed control of a nonlinear laboratory dc micro-motor using low-cost embedded control hardware and software. In *2018 5th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 1029–1034, April 2018.

APPENDIX

A. Model additivity and homogenity

For proving that additivity does not hold for the proposed model we start from Eq. (4) and make clear in the notation that T is a function of $u(t)$ so we get

$$\tau \dot{T}\left(u(t)\right) + T\left(u(t)\right) = Ku(t) + \alpha. \tag{11}$$

Assuming that the input $u_1(t)+u_2(t)$ enters the system we get

$$\begin{aligned}\tau \dot{T}\left(u_1(t)+u_2(t)\right) + T\left(u_1(t)+u_2(t)\right) = \\ = K(u_1(t)+u_2(t)) + \alpha,\end{aligned} \tag{12}$$

while the effect of the individual inputs on the output is

$$\tau \dot{T}\left(u_1(t)\right) + T\left(u_1(t)\right) = Ku_1(t) + \alpha, \tag{13}$$

$$\tau \dot{T}\left(u_2(t)\right) + T\left(u_2(t)\right) = Ku_2(t) + \alpha. \tag{14}$$

Adding the contribution of the two inputs yields

$$\begin{aligned}\tau\left(\dot{T}\left(u_1(t)\right) + \dot{T}\left(u_2(t)\right)\right) + \left(T\left(u_1(t)\right) + T\left(u_2(t)\right)\right) = \\ = K\left(u_1(t)+u_2(t)\right) + 2\alpha,\end{aligned} \tag{15}$$

from which it holds that

$$\begin{aligned}\tau\left(\dot{T}\left(u_1(t)\right) + \dot{T}\left(u_2(t)\right)\right) + \left(T\left(u_1(t)\right) + T\left(u_2(t)\right)\right) \neq \\ \neq \tau \dot{T}\left(u_1(t)+u_2(t)\right) + T\left(u_1(t)+u_2(t)\right),\end{aligned} \tag{16}$$

because

$$K\left(u_1(t)+u_2(t)\right) + \alpha \neq K\left(u_1(t)+u_2(t)\right) + 2\alpha. \tag{17}$$

For proving that homogenity does not hold for the proposed model we start with Eq. (11) and compute

$$\tau \dot{T}\left(au(t)\right) + T\left(au(t)\right) = Kau(t) + \alpha \tag{18}$$

and

$$a\left(\tau \dot{T}\left(au(t)\right) + T\left(au(t)\right)\right) = a(Ku(t) + \alpha). \tag{19}$$

Since

$$Kau(t) + \alpha \neq a(Ku(t) + \alpha), \tag{20}$$

then

$$a\left(\tau \dot{T}\left(au(t)\right) + T\left(au(t)\right)\right) \neq \tau \dot{T}\left(au(t)\right) + T\left(au(t)\right). \tag{21}$$